ELECTRONIC MODEL SHOP

DORADO MIDAS REFERENCE MANUAL

May 23, 1985

by

Frank Vest

Xerox Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA. 94304

Bravo files stored on: [Indigo]<DoradoSource>GarageMidasManual.Dm

Press files stored on: [Indigo]<DoradoDocs>GarageMidasManual.Press

This manual describes the Midas Command files used by the Electronic Model Shop to debug and test new Dorado computer systems.

TABLE OF CONTENTS

SAMPLE MIDAS DISPLAYS

MAKING AND CHANGING MIDAS COMMAND FILES

DEBUGGING

l. Introduction

This is a description about the Dorado Midas command files used at the Electronic Model shop.

The basic idea is to have the microdiagnostics load and run to completion with as simple a user interface as possible. This also makes it possible to make more complicated command files to call all of the microdiagnostics and Midas test features for verification purposes.

Several of the command files used call other command files. This is called "nested command files". When using this type of command files it is sometimes hard to get control away from the Alto computer, you bug abort with the mouse and all that happens is the current routine stops and goes on to the next routine. There is a special feature in Midas for this problem. Hold down the "**CTRL** key then "**Z**" key on the Alto keyboard.

If you would like to have the diagnostic load and wait for an input to start the diagnostic for debugging purposes see the section on changing Midas command files (page 26).

To get a Electronic Model Shop Midas disk:
1. Install a newOS on an Alto scratch disk and ask for the long installation dialogue.
2. Retrieve the command file: **[IO]<Vest>DoradoMidasDisk.cm**.
3. Type: **@DoradoMidasDisk.cm** C/R

To get a new version of the Midas command files type: **@GetGarageMidas.cm** C/R.

A copy of the disk is stored at: **[IO]<EMS>DoradoMidasDisk.Disk.**

2 Kernel.midas

The Kernel microdiagnostic will exercise all of the functionality of the Dorado processor that is not associated with the IFU, Memory, or with IO devices.

Boards needed to run Kernel: Base, ContB, ContA, ProcL, and ProcH.

Run time: About 1 second.

Actions to run Kernel: Left mouse button **RunProg**, then Left mouse button **KERNEL**.

Two things were added to the original Kernel.Midas:

1. After loading the diagnostic turn-on the **"Task-Circulate"** bit.
2. Start the program at the label Begin.

Note:
There a several different task specific items on the processor boards, R memory, T registers, etc. The first pass of Kernel is run at task level "0". After a successful completion the program will stop at the label "Done", at this point type in **;K** (to remove the breakpoint at "Done"), and **;P** (Proceed). To test all of the different task levels you need to let the program run for at least 20 seconds.

3. MemA.Midas

MemA contains diagnostic programs for each memory board in the Dorado. MemA is just a name that used to indicate that All the memory system gets tested by the diagnostic.

Boards needed to run MemA: Base, ContB, ContA, ProcL, ProcH. IFU, MemC, MemX, MemD, and two MSA boards.

Note: Jumper pin#131 left sidepanel on the IFU board to ground if you would like to run the MemA diagnostic without the IFU board installed.

Run time: About 6 minutes and 4 seconds with 1 mega-word of memory.

Actions to run MemA: Left mouse button **RunProg**, then Left mouse button **MEMA**.

Three things were added to the original MemA.Midas:

1. Reset the Droado.
2. Remove the breakpoint at "SvaTryNextPat".
3. Start the program at begin.

4. MemMisc.Midas

MemMisc checks some parts of the memory system that were later added and would not fit into MemA. You need to run MemA and MemMisc to check-out the entire memory system.

Boards needed to run MemMisc: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and two MSA boards.

Note: Jumper pin#131 left sidepanel on the IFU board to ground if you would like to run the MemMisc diagnostic without the IFU board installed.

Run time: About 5 seconds with 1 mega word of memory.

Actions to run MemMisc: Left mouse button **RunProg**, then Left mouse button **MEMMISC**.

Two things were added to the original MemMisc.Midas:

1. Reset the Dorado.
2. Start the program at begin.

5. IfuSimple.Midas

IfuSimple checks out the IFU board mostly in the single step mode, it is very useful in debugging new boards because there is no need to have the memory system installed.

Boards needed to run IfuSimple: Base, ContB, ContA, ProcL, ProcH, and IFU.

Run time: About 2 seconds.

Actions to run IfuSimple: Left mouse button **RunProg**, then Left mouse button **IFUSIMPLE**.

Two things were added to the original IfuSimple.Midas:

1. Reset the Dorado.
2. Start the program at begin.

6. IfuComplex.Midas

IfuComplex checks the IFU board at real computer speed using all of the memory system. All of the microdiagnostics should run error free before running IfuComplex.

Boards needed to run IfuComplex: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and two MSA boards.

Run time: About 7 seconds.

Actions to run IfuComplex: Left mouse button **RunProg**, then Left mouse button **IFUCOMPLEX**.

Two things were added to the original IfuComplex.Midas:

1. Reset the Dorado.
2. Start the program at begin.

7. EventCounters.Midas

EventCounters checks out the eventcounters on the IFU board. It requires a jumper installed on the right side of the Dorado. The jumper block called GenIn/GenOut is about 3 inches long and is installed on the right side of the IFU board pins #28 thru #90.

Boards needed to run EventCounters: Base, ContB, ContA, ProcL, ProcH, and IFU.

Run time: About 1 second.

Actions to run EventCounters: Left mouse button **RunProg**, then Left mouse button **EVENTCOUNTERS**.

Two things were added to the original EventCounters.Midas:

1. Reset the Dorado.
2. Start the program at begin.

8. TriconD.midas

TriconD tests the interface between the processor and the DSK/Eth board. Some of the basic data paths on the DSK/ETH board are also checked. To get the most from the test it is best to have a scratch disk up and spinning.

The program does not write on the disk or use the actual Ethernet.

Boards needed to run TriconD: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, Dsk/Eth, and two MSA boards.

Run time: About 2 seconds.

Actions to run TriconD: Left mouse button **RunProg**, then Left mouse button **TRICOND**.

Two things were added to TriconD.Midas:

1. Reset the Dorado.
2. Start the program at begin.

9. Triex.midas

Triex loads the a special version of the Alto emulation that allows the Alto program Triex to operate on a Dorado computer. In the Triex mode all of the T-80 disk drive is treated as a normal T-80, no partitions and no emulation. You should have a scratch disk installed before running Triex.

This program has a similiar look of TriconD. After loading the program from the Alto midas disk you will run the program from the Dorado display.

Boards needed to run Triex: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, Dsk/Eth, DispY and two MSA boards.

Run time: As long as you want.

Actions to run Triex: Type **Triex** to the Alto Midas display, then left mouse button **RunProg**.

10. KernelS.Midas

KernelS.Midas (speed test) first runs the microdiagnostic Kernel at 31 ns. After successful completion of one pass the clock speed is increased by one nano-second and the test is run again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored. This is due to a known problem. The command file will continue to speed up the Dorado until kernel fails. A normal Dorado will always run faster than 27 ns.

Boards needed to run KernelS: Base, ContB, ContA, ProcL, and ProcH.

Run time: About 1 second for each successful pass.

Actions to run KernelS: Type **KernelS** to the Alto Midas display, then left mouse button **RunProg**.

11. MemAS.Midas

MemAS.Midas (speed test) first runs the microdiagnostic MemA at 31 ns. After successful completion of one pass the clock speed is increased by one nano-second and the test is run again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored. This is due to a known problem. The command file will continue to speed up the Dorado until

MemA fails. A normal Dorado will always run faster than 28 ns. Since MemA requires about 6 minutes to make a full pass it should take a while before a failure happens.

Boards needed to run MemAS: Base, ContB, ContA, ProcL, Proch, IFU, MemC, MemX, MemD, and two MSA boards.

Run time: About 6 minutes for each successful pass.

Actions to run MemAS: Type **MEMAS** to the Alto Midas display, then left mouse button **RunProg**.

12. MemMiscS.Midas

MemMiscS.Midas (speed test) first runs the microdiagnostic MemMisc at 31 ns. After successful completion of one pass the clock speed is increased by one nano-second and the test is run again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored. This is due to a known problem. The command file will continue to speed up the Dorado until MemMisc fails. A normal Dorado will always run faster than 28 ns.

Boards needed to run MemMiscS: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and two MSA boards.

Run time: About 5 seconds for each successful pass.

Actions to run MemMiscS: Type **MemMiscS** to the Alto Midas display, then left mouse button **RunProg**.

13. IfuSimpleS.Midas

IfuSimpleS.Midas (speed test) first runs the microdiagnostic IfuSimple at 31 ns. After successful completion of one pass the clock speed is increased by one nano-second and the test is run again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored. This is due to a known problem. The command file will continue to speed up the Dorado until IfuSimple fails. A normal Dorado will always run faster than 28 ns.

Boards needed to run IfuSimpleS: Base, ContB, ContA, ProcL, ProcH, and IFU.

Run time: About 2 seconds for each successful pass.

Actions to run IfuSimpleS: Type **IfuSimpleS** to the Alto Midas display, then left mouse button **RunProg**.

14. IfuComplexS.Midas

IfuComplexS.Midas (speed test) first runs the microdiagnostic IfuComplex at 31 ns. After successful completion of one pass the clock speed is increased by one nano-second and the test is run again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored. This is due to a known hardware problem. The command file will continue to speed up the Dorado until IfuComplex fails. A normal Dorado will always run faster than 28 ns.

Boards needed to run IfuComplexS: MemA: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and two MSA boards.

Run time: About 7 seconds for each successful pass.

Actions to run IfuComplexS: Type **IfuComplexS** to the Alto Midas display, then left mouse button

RunProg.

15. SboardTest.Midas

This Midas command file first loads MemA and then changes MemA to run the Storage board tests only. This is helpful for working on a known bad MSA board.

Boards needed to run SboardTest: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and two MSA boards.

Run time: About 5 minutes with 1 mega-word of memory.

Actions to run Sboardtest: Left mouse button **RunProg**, then Left mouse button **SBOARDTEST**.

16. CboardTest.Midas

This Midas command file first loads MemA and then changes MemA to run the MemC board tests only. This is helpful for working on a known bad MemC board.

Boards needed to run CboardTest: Base, ContB, ContA, ProcL, ProcH, IFU, and MemC.

A special jumper on the side panel is required to run this test if the boards above the MemC are removed: **Left sidepanel MemC board pin#122 to ground.**

Run time: About 10 seconds.

Actions to run Cboardtest: Left mouse button **RunProg**, then Left mouse button **CBOARDTEST**.

17. XboardTest.Midas

This Midas command file first loads MemA and then changes MemA to run the MemX board tests only. This is helpful for working on a known bad MemX board.

Boards needed to run XboardTest: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, and MemX.

Run time: About 15 seconds.

Actions to run Xboardtest: Left mouse button **RunProg**, then Left mouse button **XBOARDTEST**.

18. DboardTest.Midas

This Midas command file first loads MemA and then changes MemA to run the MemD board tests only. This is helpful for working on a known bad MemD board.

Boards needed to run DboardTest: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, and MemD.

Run time: About 33 seconds.

Actions to run Dboardtest: Left mouse button **RunProg**, then Left mouse button **DBOARDTEST**.

19. FIO.Midas

The FIO (**F**ast **I**nput/**O**utput) test has to have the FIO test board installed in one of the fast I/O

slots before the test will run.

This command file loads the MemMisc microdiagnostic and turns on the Fast Input/Output part of MemMisc.

Boards needed to run FIO: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, two MSA boards and the FIO test board.

Run time: About 7 seconds.

Actions to run FIO: Left mouse button **RunProg**, then Left mouse button **FIO**.

20. LAG.Midas

LAG.Midas (**L**oad **A**nd **G**o) continously runs the microdiagnostics Kernel, MemMisc, IfuSimple, IfuComplex, EventCounters, and MemA. The speed of the Dorado is not changed by this Command file, so you can run LAG at any speed you desire.

Boards needed to run LAG: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and two MSA boards.

Run time: About 8 minutes for each pass with 1 mega-word of memory.

Actions to run LAG: Left mouse button **RunProg**, then Left mouse button **LAG**.

21. LAG1.Midas

LAG1.Midas (Load And Go 1 time) runs the microdiagnostics Kernel, MemMisc, IfuSimple, IfuComplex, EventCounters, and MemA one time at a speed of 30 ns.

Boards needed to run LAG1: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and two MSA boards.

Run time: About 10 minutes with 1 mega-word of memory.
Actions to run LAG1: Left mouse button **RunProg**, then Left mouse button **LAG1**.

22. LAGSpeedTest.Midas

LAGSpeedTest runs the microdiagnostics Kernel, MemMisc, IfuSimple, IfuComplex, EventCounters, and MemA at a speed of 31 ns. After a successful completion changes the clock speed one nano second faster and runs again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored, this is due to a known hardware problem.

Boards needed to run LagSpeedTest: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and two MSA boards.

Run time: About 10 minutes for each pass with 1 mega word of memory.

Actions to run LAGSpeedTest: Left mouse button **RunProg**, then Left mouse button **LAGSPEEDTEST**.

23 KernelTask.Midas

This command file runs the Kernel microdiagnostic to completion with task circulation turned on.

After a successful pass, this command file checks for the proper task number. If the task number is correct it runs the test again. It will do this for 37 times, enough to go through all of the different task levels 2 times.

Boards needed to run KernelTask: Base, ContB, ContA, ProcL, and ProcH.

Run time: About 30 seconds.

Actions to run KernelTask: Type **KernelTask** to the Alto Midas display, then left mouse button **RunProg**.

24. VoltageTest.Midas

The Voltage test reads the 4 voltages from the baseboard and compares them for upper and lower limits. There are 3 places on the Baseboard where the voltages are tested: VOLTS, MINVOLTS, and MAXVOLTS.

Note: There is no Midas message for a successful pass of this test.

Sample Error message:
MAXVOLTS -2 Volts too high (2.10 max.)

The limits for each voltage are:

+5 volts	4.82-5.19
+12 volts	11.51-12.51
-5.2 volts	5.05-5.36
-2 volts	1.89-2.10

Boards needed to run Voltagetest: Baseboard.

Run time: About 5 seconds.

Actions to run VoltageTest: Type **VoltageTest** to the Alto Midas display, then left mouse button **RdCmds**.

25. CurrentTest.Midas

The CurrentTest reads the 4 currents from the baseboard and compares them for upper and lower limits. There are 3 places on the Baseboard where the currents are tested: AMPS, MINAMPS, and MAXAMPS.

Note: There is no Midas message for a successful pass of this test.

The limits for each current is:

+5 volts	0-61 amps
+12 volts	0-3 amps
-5.2 volts	151-222 amps
-2 volts	41-69 amps

Boards needed to run CurrentTest: This command file is designed to run with a full running Dorado.

Run time: About 5 seconds.

Actions to run CurrentTest: Type **CurrentTest** to the Alto Midas display, then left mouse button

RdCmds.

26. TemperatureTest.Midas

The TemperatureTest reads the temperature sensors on all of the boards that are supposed to have them installed. The test range is set for a minimum of 17 degrees C. and a maximum of 41 degrees C.. There are 6 places on the Baseboard where the temperatures are tested: TEMP0, TEMP0+1, TEMP0+2, MAXTEMP0, MAXTEMP0+1, and MAXTEMP0+2.

Note: There is no Midas message for a successful pass of this test.

Boards needed to run TemperatureTest: This command file is designed to run with a full running Dorado.

Run time: About 6 seconds.

Actions to run TemperatureTest: Type **TemperatureTest** to the Alto Midas display, then left mouse button **RdCmds**..

27. VIT.Midas

VIT.Midas is a command file that runs the VoltageTest, CurrentTest, and the TemperatureTest.

Note: There is no Midas message for a successful pass of this test.

Boards needed to run VIT: This command file is designed to run with a full running Dorado.

Run time: About 16 seconds.

Actions to run VIT: Left mouse button **RdCmds**, then Left mouse button **VIT**.

28. TestAllGarage.Midas

TestAllGarage tests all of the memories and registers available in the **"Test"** function of Midas. The normal **"TestAll"** function leaves out 5 things: IMBD, MAP, VM, IFUM, and ESTAT.

In addition to testing more registers and memories comments were added to help in diagnosing the failure.

Sample Error message:
"INSSET" FAILED (INSTRUCTION SET) PG. 4 IFU BOARD.

Boards needed to run TestAllGarage: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and 2 MSA boards.

Run time: About 10 minutes.

Actions to run TestAllGarage: Type **TestAllGarage** to the Alto Midas display, then left mouse button **RdCmds**.

29. AcceptanceTests.Midas

AcceptanceTests sets the clocks to 30 nano-seconds and then runs VoltageTest, CurrentTest, TemperatureTest, TestAllGarage, Kernel, KernelTask, MemMisc, IfuSimple, IfuComplex,

EventCounters, MemA and TriconD.

After successful completion of the Microdiagnostics a 2 minute simtest is run, if there are any errors they are logged on the file "Simerror.report" and the test is restarted. The test will stop logging errors after 5 SimTest errors. You can go into Bravo and look at the errors or print the errors on your favorite printer.

"Note" If there is an error and you want to abort the AcceptanceTests command file, hold down the **CTRL** key then push the "**Z**" key on the Alto keyboard. This is a special feature of Midas that will kill a "nested command file". Another way to kill the test is with the boot button.

Boards needed to run AcceptanceTests.Midas: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, Dsk/Eth, and two MSA boards.

Run time: About 22.5 minutes and with 1 mega-word of memory.
NOTE: 2 MSA boards with 64k chips = 1 mega-word of memory.

Actions to run AcceptnaceTests: Left mouse button **RdCmds**, then Left mouse button **ACCEPTANCETESTS**.

30. MapAddr.Midas

The Dorado Map has 177777 octal locations. This requires 16 address bits. MapAddr.Midas loads on the center column of the midas display 16 different map locations, one entry for each of the 16 different address bits. After all of the Map locations are displayed, each location is then loaded and checked with a value equal to its own address. Map location 100 would have a value of 100. The Map is located on the MemX board.

Center column of the Midas display after running MapAddr.Midas

```
MAP 0              0
MAP 1              1
MAP 2              2
MAP 4              4
MAP 10            10
MAP 20            20
MAP 40            40
MAP 100          100
MAP 200          200
MAP 400          400
MAP 1000        1000
MAP 2000        2000
MAP 4000        4000
MAP 10000      10000
MAP 20000      20000
MAP 40000      40000
MAP 100000    100000
```

Boards needed to run MapAddr.Midas: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, and MemD.

Run time: About 7 seconds.

Actions to run MapAddr: Type **MapAddr** on the Alto keyboard, then Left mouse button **RdCmds**.

31. CacheDAddr.Midas

The Dorado CacheD (**Cache D**ata) has 7777 octal locations. This requires 12 address bits. CacheDAddr.Midas loads on the center column of the midas display 12 different CacheD locations, one entry for each of the 12 different address bits. After all of the CacheD locations are displayed, each location is then loaded and checked with a value equal to its own address. CacheD location 100 would have a value of 100. The CacheD is located on the MemD board.

Center column of the Midas display after running CacheDAddr.Midas

```
CACHED 0              0
CACHED 1              1
CACHED 2              2
CACHED 4              4
CACHED 10            10
CACHED 20            20
CACHED 40            40
CACHED 100          100
CACHED 200          200
CACHED 400          400
CACHED 1000        1000
CACHED 2000        2000
CACHED 4000        4000
CACHED 6000        6000
```

Boards needed to run CacheDAddr.Midas: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX and MemD.

Run time: About 17 seconds.

Actions to run CacheDAddr: Type **CacheDAddr** on the Alto keyboard, then Left mouse button **RdCmds**.

32. CacheAAddr.Midas

The Dorado CacheA (**Cache A**ddress Memory) has 377 octal locations. This requires 8 address bits. CacheAAddr.Midas loads on the center column of the midas display 8 different CacheA locations, one entry for each of the 8 different address bits. After all of the CacheA locations are displayed, each location is then loaded and checked with a value equal to its own address. CacheA location 100 would have a value of 100.

Center column of the Midas display after running CacheDAddr.Midas

```
CACHEA 0              0
CACHEA 1              1
CACHEA 2              2
CACHEA 4              4
CACHEA 10            10
CACHEA 20            20
CACHEA 40            40
CACHEA 100          100
CACHEA 200          200
```

Boards needed to run CacheAAddr.Midas: Base, ContB, ContA, ProcL, ProcH, IFU, and MemC.

Run time: About 6 seconds.

Actions to run CacheAAddr: Type **CacheAAddr** on the Alto keyboard, then Left mouse button **RdCmds**.

33. StkAddr.Midas

The Dorado **St**ac**k** has 377 octal locations. This requires 8 address bits. StkAAddr.Midas loads on the center column of the midas display 8 different Stack locations, one entry for each of the 8 different address bits. After all of the Stack locations are displayed, each location is then loaded and checked with a value equal to its own address. Stack location 100 would have a value of 100. The Stack memory is located on the ProcH and ProcL boards.

Center column of the Midas display after running StkAddr.Midas

```
STK 0              0
STK 1              1
STK 2              2
STK 4              4
STK 10            10
STK 20            20
STK 40            40
STK 100          100
STK 200          200
```

Boards needed to run StkAddr.Midas: Base, ContB, ContA, ProcL, and ProcH.

Run time: About 4 seconds.

Actions to run StkAddr: Type **StkAddr** on the Alto keyboard, then Left mouse button **RdCmds**.

34. RMAddr.Midas

The Dorado RM (**R**egister **M**emory) has 377 octal locations. This requires 8 address bits. RMAddr.Midas loads on the center column of the midas display 8 different RM locations, one entry for each of the 8 different address bits. After all of the RM locations are displayed, each location is then loaded and checked with a value equal to its own address. RM location 100 would have a value of 100. The R memory is located on the ProcH and ProcL boards.

Center column of the Midas display after running RMAddr.Midas

```
RM 0               0
RM 1               1
RM 2               2
RM 4               4
RM 10             10
RM 20             20
RM 40             40
RM 100           100
RM 200           200
```

Boards needed to run RMAddr.Midas: Base, ContB, ContA, ProcL, and ProcH.

Run time: About 3 seconds.

Actions to run RMAddr: Type **RMAddr** on the Alto keyboard, then Left mouse button **RdCmds**.

35. IMXAddr.Midas

The Dorado IMX (Instruction Memory) has 7777 octal locations. This requires 12 address bits. IMXAddr.Midas loads on the right column of the midas display 12 different IMX locations, one entry for each of the 12 different address bits. After all of the IMX locations are displayed, each location is then loaded and checked with a value equal to its own address. IMX location 100 would have a value of 100. The IMX is located on the ContB board.

Boards needed to run IMXAddr.Midas: Base, ContB, ContA, ProcL ,ProcH, IFU, MemC, and MemX.

Run time: About 3 seconds.

Actions to run IMXAddr: Type **IMXAddr** on the Alto keyboard then Left mouse button **RdCmds**.

36. IFUMAddr.Midas

The Dorado IFU memory has 1777 octal locations. This requires 10 address bits. IFUMAddr.Midas loads on the center column of the midas display 10 different IFUM locations, one entry for each of the 10 different address bits. After all of the IFUM locations are displayed, each location is then loaded and checked with a value equal to its own address. IFUM location 100 would have a value of 100.

Boards needed to run IFUMAddr.Midas: Base, ContB, ContA, ProcL, ProcH, and IFU.

Run time: About 4 seconds.

Actions to run IFUMAddr: Type **IFUMAddr** on the Alto keyboard, then Left mouse button **RdCmds**.

37. BRAddr.Midas

The Dorado memory **B**ase **R**egister has 37 octal locations. This requires 5 address bits. BRAddr.Midas loads on the center column of the midas display 5 different BR locations, one entry for each of the 5 different address bits. After all of the BR locations are displayed, each location is then loaded and checked with a value equal to its own address. BR location 10 would have a value of 10. The memory base register is located on the MemC board with the addressing coming from the ProcH board.

Boards needed to run BRAddr.Midas: Base, ContB, ContA, ProcL, ProcH, IFU, and MemC.

Run time: About 2 seconds.

Actions to run BRAddr: Type **BRAddr** on the Alto keyboard, then Left mouse button **RdCmds**.

38. Klink.Midas

This is a special command file to be used when the IFU board gets lost during the IfuComplex microdiagnostic. It is used to find a pointer to where the IFU was supposed to jump before it went off to never-never land.

Boards needed to run Klink.Midas: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and two MSA boards.

Run time: About 1 second.

Actions to run Klink: Left mouse button **RdCmds**, then Left mouse button **Klink**.

39. BadChip.Midas

This command file is only useful when there is a single bit error found when running the storage part of the MemA test. If you have a system with more than 2 MSA boards you will have to figure out which pair of MSA boards the test failed on. The command file **MsaPair.midas** will tell you which pair of MSA boards is being tested.

Boards needed to run BadChip.Midas: Base, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and two MSA boards.

Run time: Up to 1 minute if the bad chip is located at J-23 on the bottom MSA board.

Actions to run BadChip: Left mouse button **RdCmds**, then Left mouse button **BADCHIP**.

40. MSAPair.Midas

This command file will tell you which pair of MSA boards were being tested during the storage part of the MSA test. The only time this test is useful is when there are 4 or more MSA boards installed in the system. This test assumes that the MSA boards are installed starting from the bottom to the top with no holes between MSA boards.

Run time: About 1 second.

Actions to run MSAPair.Midas: Left mouse button **RdCmds**, then Left mouse button **MSAPAIR**.

41. IMRH.Midas

This is a special command file to turnoff the checking of **I**nstruction **M**emory **R**ight **H**and parity errors for debugging.

Run time: About 1 second.

Actions to run IMRH: type **IMRH** to the Alto display, then Left mouse button **RdCmds.**

42. IMLH.Midas

This is a special command file to turnoff the checking of **I**nstruction **M**emory **L**eft **H**and parity errors for debugging.

Run time: About 1 second.

Actions to run IMLH: type **IMLH** to the Alto display, then Left mouse button **RdCmds.**

43. RAMPE.Midas

This is a special command file to turnoff the checking of **R**egister **M**emory parity errors for

debugging.

Run time: About 1 second.

Actions to run RAMPE: type **RAMPE** to the Alto display then Left mouse button **RdCmds.**

44. IOBPE.Midas

This is a special command file to turnoff the checking of **I**nput **O**utput **B**uss **P**arity **E**rrors for debugging.

Run time: About 1 second.

Actions to run IOBPE: type **IOBPE** to the Alto display, then Left mouse button **RdCmds.**

45. MDPE.Midas

This is a special command file to turnoff the checking of **M**emory **D**ata **P**arity **E**rrors for debugging.

Run time: About 1 second.

Actions to run MDPE: type **MDPE** to the Alto display, then Left mouse button **RdCmds.**

46. MemoryPE.Midas

This is a special command file to turnoff the checking of **M**emory **P**arity **E**rrors for debugging.

Run time: About 1 second.

Actions to run MemoryPE: type **MemoryPE** to the Alto display then, Left mouse button **RdCmds.**

47. SaveDisplay.Midas

This command file will save the Alto Midas display on a file called **Crash.report**. You can look at the file in bravo or print it on your favorite printer.

Run time: About 3 seconds.

Actions to run SaveDisplay: type **SaveDisplay** to the Alto display, then Left mouse button **RdCmds.**

48. SaveIFUCrash.Midas

This command file will save the Alto Midas display plus most of the information about the IFU board on a file called **IFUCrash.report**. You can look at the file in bravo or print it on your favorite printer.

Run time: About 30 seconds.

Actions to run SaveIFUCrash: type **SaveIFUCrash** to the Alto display, then Left mouse button **RdCmds.**

49. SimTestNoErrors.Midas

This command file will turn off the checking of 6 signals that commonly fail during the running of SimTest.
The Signals turned off are: bSwitch'a, DSel0, DSel1, ColVic.0, ColVic.1, and STWati-Mem'.

Run time: Until the first **Simtest** error occurs.

Actions to run SimTestNoErrors: type **SimTestNoErrors** to the Alto display, then Left mouse button **RdCmds.**

50. SimTest20.Midas

This command file turns off all of the signal checking in SimTestNoErrors.midas and then runs the SimTest. Each error is logged on a file called SimError.Report. This will continue until 20 errors are logged. You can look at the file in bravo or print it on your favorite printer.

Run time: It could run forever.

Actions to run SimTest20: type **SimTest20** to the Alto display, then Left mouse button **RdCmds.**

51. StartMap.Midas

This command file is used to turn off the DMux checking of the signal "StartMap" from the MemC board. It is useful when running Simtest with the boards above the MemC board removed.

Another way to solve the problem with "StartMap" is to ground pin#102 on the left side of the MemC board.

Run time: 1 second.

Actions to run StartMap: type **StartMap** to the Alto display, then Left mouse button **RdCmds.**

52 MIRDebug.Midas

This is a special command file to turnoff MIRDebug.

Run time: About 1 second.

Actions to run MIRDebug: type **MIRDebug** to the Alto display, then Left mouse button **RdCmds.**

53. IMBDAddr.Midas

IMBD is a special path (DMUX) to the ContB board to check the Instruction memory (IMX) without the ContA board installed. The Dorado IMBD has 7777 octal locations. This requires 12 address bits. IMBDAddr.Midas loads on the right column of the midas display 12 different IMBD locations, one entry for each of the 12 different address bits. After all of the IMBD locations are displayed, each location is then loaded and checked with a value equal to its own address.

Boards needed to run IMBDAddr.Midas: Base and ContB.

Run time: About 3 seconds.

Actions to run IMBDAddr: Type **IMBDAddr** to the Alto, then Left mouse button **RdCmds**.

54. FMemA.Midas

FMemA (Fast MemA) is a quick test of the memory system. After running the Memc, MemX, and the MemD part of MemA, I set the pattern count to 70 for the MSA part of the tests. This removes the "Bit Slice" testing and just tests the memory for random pattern and addressing tests.

Boards needed to run FMemA: Base, ContB, ContA, ProcL, ProcH. IFU, MemC, MemX, MemD, and two MSA boards.

Run time: About 2 minutes with 1 mega-word of memory.

Actions to run FMemA: Left mouse button **RunProg**, then Left mouse button **FMEMA**.

55. CiaMap.Midas

CiaMap is helpful in figuring how the microcode is running. This command file will open an output file called CiaMap.Report, then single step the Dorado, "PrettyPrint" CIA and loop doing this forever. You will have to Shift/Swat to go to the Executive and then print the file CiaMap.Report on your favorite printer.

Run time: As long as you want, but you could easily fill up your Alto disk if you wait too long.

Actions to run CiaMap: Insert a breakpoint at the location in the microprogram where you want to start mapping. Then run the program until it hits the breakpoint and type **CIAMAP** to the Alto, then Left mouse button **RdCmds**.

Sample output from CiaMap:

CIA
SETBR = abs. 4440

CIA
SETBR+1 = abs. 4440

CIA
SETBR+2 = abs. 4440

CIA
BEGIN+6 = abs. 4440

CIA
RESETIFU = abs. 4440

CIA
RESETIFU+1 = abs. 4440

CIA
RESETIFU+2 = abs. 4440

CIA
RESETIFU+3 = abs. 4440

CIA
RESETIFU+4 = abs. 4440

CIA
RESETIFU+5 = abs. 4440

CIA
RESETIFU+6 = abs. 4440

CIA
LONGWAIT = abs. 4440

CIA
LONGWAIT+1 = abs. 4440

56. MirMap.Midas

MirMAp is helpful in figuring how the microcode is running. This command file will open an output file called MirMAp.Report, then single step the Dorado, "PrettyPrint" CIA and loop doing this forever. You will have to Shift/Swat to go to the Executive and then print the file MirMAp.Report on your favorite printer.

Run time: As long as you want, but you could easily fill up your Alto disk if you wait too long.

Actions to run MirMAp: Insert a breakpoint at the location in the microprogram where you want to start mapping. Then run the program until it hits the breakpoint and type **MIRMAP** to the Alto, then Left mouse button **RdCmds**..

Sample output from MirMap:

MIR
NEXTPAT8: LocGo[.+1], A_R0, Pd_R0, RBase_3S
RSTK=0, ALUF=0, BSEL=1, LC=0, ASEL=4, BLOCK=0, FF=203, JCN=224

MIR
NEXTPAT8+1: LocGo[.+1], R14_T_(A_R14)+1, B_R14, RBase_0S
RSTK=14, ALUF=13, BSEL=1, LC=7, ASEL=4, BLOCK=0, FF=200, JCN=244

MIR
NEXTPAT8+2: Return, R0, Pd_(A_T)-(400C)
RSTK=0, ALUF=5, BSEL=6, LC=0, ASEL=6, BLOCK=0, FF=1, JCN=107

MIR
IFUPCALPHAL+1: LocGo[.+1,.+2,ALU=0], A_R6, R6_T
RSTK=6, ALUF=0, BSEL=2, LC=6, ASEL=4, BLOCK=0, FF=60, JCN=272

MIR
IFUPCALPHAL+3: LongGo[.+1], A_R0, Pd_R0
RSTK=0, ALUF=0, BSEL=1, LC=0, ASEL=4, BLOCK=0, FF=15, JCN=11

MIR
IFUPCALPHAL+4: LongCall[PUTCDBYTE], T_(A_Stack&ChkUFL)+1, B_Stack&ChkUFL
RSTK=10, ALUF=13, BSEL=1, LC=1, ASEL=4, BLOCK=1, FF=25, JCN=0

MIR

```
PUTCDBYTE: LongGo[.+1], A_Stack&+1, Pd_Stack&+1
RSTK=1, ALUF=0, BSEL=1, LC=0, ASEL=4, BLOCK=1, FF=204, JCN=10

MIR
PUTCDBYTE+1: LocGo[.+1], A_Stack&ChkUFL, Stack&ChkUFL_Link
RSTK=10, ALUF=0, BSEL=1, LC=6, ASEL=4, BLOCK=1, FF=177, JCN=265

MIR
PUTCDBYTE+2: LongCall[FIXBYTEADDRFORINSTRSET], A_R0, Pd_R0
RSTK=0, ALUF=0, BSEL=1, LC=0, ASEL=4, BLOCK=0, FF=325, JCN=0

MIR
FIXBYTEADDRFORINSTRSET: LongGo[.+1], A_Stack&+1, Pd_Stack&+1
RSTK=1, ALUF=0, BSEL=1, LC=0, ASEL=4, BLOCK=1, FF=311, JCN=17

MIR
FIXBYTEADDRFORINSTRSET+1: LocGo[.+1], A_Stack&+1, Stack&+1_Link
RSTK=1, ALUF=0, BSEL=1, LC=6, ASEL=4, BLOCK=1, FF=177, JCN=236

MIR
FIXBYTEADDRFORINSTRSET+2: LongGo[.+1], A_Stack&ChkUFL, Stack&ChkUFL_T
RSTK=10, ALUF=0, BSEL=2, LC=6, ASEL=4, BLOCK=1, FF=220, JCN=2
```

1. Tests Display

Actions to get this display: Left mouse button **RdCmds**, then left mouse button **Tests**.

This is the display that should be put up if there is a failure during **Test**, **TestAll**, or **GarageTestAll**.

```
*CONFIG     6612   PROBLEMS           0   LOOP-COUNT                    0
*CLKRUN    37673   OUTOFSPEC          0   DATA-WAS                      0
*ESTAT     10477   BADSUPPLYSPEC      0   SHOULD-BE                     0
*INSSET      2 0                           BITS-CHECKED      3 377777 377777
*OLINK 20   2123                           BITS-PICKED                   0
*TLINK 20   2124                           BITS-DROPPED                  0
*TPC 20     2201                           NFAILURES                     0
*RBASE 20      1
*MEMBASE 20   36                           LOW-ADDR                      0
 T 20          0                           HIGH-ADDR           177777 177777
 TIOA 20       0                           CURRENT-ADDR                  0
*CNT      177777                           ADDR-INC                      1
*STKP          4                           ADDR-INTERS                   0
 MEMBX         0                           ADDR-UNION                    0
 Q             0
*SHC         2114                          COMM-ERRS                     0
*PCX            6                          MIR-PES                       0
 PROCSRN       0                           DWATCH                        0
*MCR         6002                          IMOUT               204364 225323
 TASK          0                           MIR                 261265 034107

Midas 6/28/83, D1Midas 6/28/83       Serial #377   Time: 03.58


RunProg RdCmds Dtach Exit Brk UnBrk Go SS OS Ld LdSyms Cmpr Reset SetClk
Config PEscan TestAll Test SimTest SimGo HWchk T1 T2 T3 RepGo RepSS RepT2
Fields LDRtest ShowCmds WrtCmds Abs Active DMux
```

The "Midas" display is made up of 3 columns: left, middle, and right. The left column is referred to as the "A" column, the center column is called the "B" column and the right column is called the "C" column.

2. mmc.Midas

Actions to get this display: Left mouse button **RdCmds**, then left mouse button **mmc**.

This is the display that can be used to help in the debugging of the MemC board. It puts on the display interesting things about the MemC board.

```
*CONFIG     6612   PROBLEMS           0  *UPTIME             0 days 0:42:33
*CLKRUN    37673   OUTOFSPEC          0   TGLITCH              0 days 0:0:0
*ESTAT     10477   BADSUPPLYSPEC      0   COMM-ERRS                       0
*INSSET      2 0                          MIR-PES                         0
*OLINK 20   2123   PVAH             200  *VOLTS    +12.07 +5.03 -2.01 -5.28
*TLINK 20   2124   PVAL               0  *AMPS          0    13    54   181
*TPC 20     2201   MAPAD             20  *TEMP0       +25   +31    ??   +25
*RBASE 20      1   HIT              310  *TEMP0+1     +33   +25    ??    ??
*MEMBASE 20   36   HOLD           37013  *TEMP0+2     +31   +29    --    --
 T 20           0   PAIR           21063  *MINVOLTS +12.07 +5.03 -2.00 -5.26
 TIOA 20        0   PIPEAD          7400  *MAXVOLTS +12.07 +5.06 -2.02 -5.28
*CNT       177777   AAD                0  *MINAMPS       0     3    48   181
*STKP           4   MEMB              37  *MAXAMPS       0    27    59   189
 MEMBX          0                        *MAXTEMP0    +27   +31    ??   +25
 Q              0   MAR                0  *MAXTEMP0+1  +33   +25    ??    ??
*SHC         2114   BMUX               0  *MAXTEMP0+2  +31   +29    ??    ??
*PCX            6
 PROCSRN        0   PIPE 0          62510   DWATCH                          0
*MCR         6002            40160 000051  *IMOUT               204364 225323
 TASK           0           111400 014411  *MIR                  261265 034107

Midas 6/28/83, D1Midas 6/28/83       Serial #377   Time: 02.59


RunProg RdCmds Dtach Exit Brk UnBrk Go SS OS Ld LdSyms Cmpr Reset SetClk
Config PEscan TestAll Test SimTest SimGo HWchk T1 T2 T3 RepGo RepSS RepT2
Fields LDRtest ShowCmds WrtCmds Abs Active DMux
```

3. MapAddr.Midas Display

The Dorado map has 64k locations. This requires 16 address bits. MapAddr.Midas loads on the center column of the midas display 16 different map locations, one entry for each of the 16 different address bits. After the display has all of the map locations displayed each location is then loaded with its own address. Map location 100 would have a value of 100.

Actions to get this display: Left mouse button **RdCmds**, then left mouse button **MapAddr**.

```
CONFIG     6612   MAP 0     2 0 000000  *UPTIME             0 days 0:40:31
CLKRUN    37673   MAP 1               1   TGLITCH             0 days 0:0:0
ESTAT     10477   MAP 2               2   COMM-ERRS                      0
INSSET        2 0 MAP 4               4   MIR-PES                        0
OLINK 20   2123   MAP 10             10  *VOLTS    +12.07 +5.03 -2.01 -5.28
TLINK 20   2124   MAP 20             20  *AMPS          0    11    54   181
TPC 20     2201   MAP 40             40  *TEMP0       +25   +31    ??   +25
RBASE 20      1   MAP 100           100   TEMP0+1     +33   +25    ??    ??
MEMBASE 20   36   MAP 200           200   TEMP0+2     +31   +29    --    --
T 20          0   MAP 400           400   MINVOLTS +12.07 +5.03 -2.00 -5.26
TIOA 20       0   MAP 1000         1000   MAXVOLTS +12.07 +5.06 -2.02 -5.28
CNT      177777   MAP 2000         2000   MINAMPS       0     3    48   181
STKP          4   MAP 4000         4000   MAXAMPS       0    27    59   189
MEMBX         0   MAP 10000       10000   MAXTEMP0    +27   +31    ??   +25
Q             0   MAP 20000       20000   MAXTEMP0+1  +33   +25    ??    ??
SHC        2114   MAP 40000       40000   MAXTEMP0+2  +31   +29    ??    ??
PCX           6  *MAP 100000     100000
PROCSRN       0                           DWATCH                         0
MCR        6002                           IMOUT                 204364 225323
TASK          0                           MIR                   261265 034107

                                      Serial #377   Time: 06.69


RunProg RdCmds Dtach Exit Brk UnBrk Go SS OS Ld LdSyms Cmpr Reset SetClk
Config PEscan TestAll Test SimTest SimGo HWchk T1 T2 T3 RepGo RepSS RepT2
Fields LDRtest ShowCmds WrtCmds Abs Active DMux
```

4. Midas display with error from the Map

```
CONFIG     3611   MAP 0     3 0 004000   LOOP-COUNT                  366
CLKRUN    37673   MAP 1     1 0 004001   DATA-WAS              3 0 004000
ESTAT     10477   MAP 2     1 0 004002   SHOULD-BE             3 0 000000
INSSET        0   MAP 4     1 0 004004   BITS-CHECKED            3 177777
OLINK 20   6350   MAP 10    1 4 004010   BITS-PICKED                4000
TLINK 20   6351   MAP 20    1 0 004020   BITS-DROPPED                  0
TPC 20     6106   MAP 40    1 0 004040   NFAILURES                     0
RBASE 20      0   MAP 100   1 0 004100
MEMBASE 20    0   MAP 200   1 0 004200   LOW-ADDR                      0
T 20     177777   MAP 400   1 0 004400   HIGH-ADDR                177777
TIOA 20       0   MAP 1000  1 0 005000   CURRENT-ADDR                  1
CNT       45464   MAP 2000  1 0 006000   ADDR-INC                      1
STKP        132   MAP 4000        4000   ADDR-INTERS          177777 177777
MEMBX         2   MAP 10000 1 0 014000   ADDR-UNION                    0
Q        132343   MAP 20000 1 0 024000
SHC       32001   MAP 40000 1 0 044000   COMM-ERRS                     0
PCX      170726   MAP 1000001 0 104000   MIR-PES                       0
PROCSRN       0                          DWATCH                        0
MCR       11001                          IMOUT                100164 077612
TASK         17                          MIR                     204 017371

                                    Serial #377   Time: 00.23

WP Dirty RP[0:15]=177777


RunProg RdCmds Dtach Exit Brk UnBrk Go SS OS Ld LdSyms Cmpr Reset SetClk
Config PEscan TestAll Test SimTest SimGo HWchk T1 T2 T3 RepGo RepSS RepT2
Fields LDRtest ShowCmds WrtCmds Abs Active DMux
```

The "Midas" display is made up of 3 columns: left, middle, and right. The left column is referred to as the "A" column, the center column is called the "B" column and the right column is called the "C" column.

On this display we can see a bit being picked by the memory map. The memory map is located on the MemX board. In the "B" column we see some selected locations displayed using the command file "MapAddr.midas", all of them have bit #4 stuck on. In the "C" column we see the "DATA-WAS" and "SHOULD-BE" information. Also in the "C" column there is an entry for "BITS-PICKED" and one for "BITS-DROPPED". This shows bit#4 as the bit picked.

1. Making Midas command files using WrtCmds

The **WrtCmds** is a very useful command to use if you are making your own command files.

If you wanted to make a command file to loop on a failure first type in a name for your command file, "**FOO**" is a popular name. Then left mouse button **WrtCmds .** You now do everything you would normally do to put a microdiagnostic in a loop. Every mouse command or type-in is recorded in the command file. After you are finished with what you wanted to do, left mouse button **StopWrt** and your command file is finished.

This is a sample of a command file designed to loop on an error, you would probably go into Bravo and increase the **TimeOut** to a very large number so the command file would run longer than 12 seconds. You would also want to remove the **Abort** from the command file.

```
L C9 Addr APLUS1
MR C9 FillC APLUS1
L C12 Val (LONGGO[APLUS1]
L C13 Val (LONGGO[APLUS1]
L X TimeOut 30000; 12 sec
L X Go BEGIN; ;G
L X Skip 1
L X ShowError Timed out
L X Abort BEGIN; control-C
```

This is an example of a command file after editing using Bravo. The **TimeOut** is increased and the **Abort** is removed.

```
L C9 Addr APLUS1
MR C9 FillC APLUS1
L C12 Val (LONGGO[APLUS1]
L C13 Val (LONGGO[APLUS1]
L X TimeOut 17777777; A very long timeout
L X Go BEGIN; ;G
L X Skip 1
L X ShowError Timed out
```

To run the command file: type **FOO** to the Midas Display and left mouse button **RdCmds**.

This might seem like a lot of work but setup time and correctness of the design of a scope loop is very important. Once a command file is made it is very simple process to return the machine to exactly the same condition.

2. Changing Midas command files using Bravo

Using bravo, insert a semicolon to comment out things you don't want to happen.

Example of Midas command file with the "Task circulation" turned off .

```
L X Reset                                   ; KERNEL.MIDAS display relevant rm, im values for kernel
diagnostics
L X Do-it
L X Ld KERNEL                   ; load the microprogram
L B0 Addr RBASE 0
L B1 Addr RBASE 17
L B0 Val 0
L B1 Val 0
```

```
L B2 Addr MCR
L B2 Val  1              ; turn off stack overflow/underflow wakeups from memC
L B4 Addr STACKPTOPBITS
L B5 Addr STACKPADDR
L C0 Addr R0                    ; display common registers
L C1 Addr R1
L C2 Addr RM1
L C3 Addr R01
L C4 Addr R10
L C5 Addr RHIGH1
L C6 Addr RSCR
L C7 Addr RSCR2
L C8 Addr T 20
L C9 Addr FLAGS                 ; control for  hold, task simulator
L C10 Addr ITERATIONS           ; count of iterations
L C11 Addr NEXTTASK
L C12 Addr HOLDVALUE
L C13 Addr
L C14 Addr
L A19 Addr TASK 20    ; use task 0 as default
L A19 Val 0
L A7 Addr RBASE 20    ; use rbase 17 as default
L A7 Val 0
L X DisplayOn                           ; May 18, 1981  11:43 AM
;L X TimeOut 10000
:L X Call XORTASKCIRC()                 ; TURN ON TASK CIRCULATE
;L X Skip 1
;L X ShowError Timed out
L X TimeOut 100000
L X Go BEGIN
L X Skip 1
L X ShowError Timed out
```

Example of Midas command file with the "Go at Begin" removed.

```
L X Reset                               ; KERNEL.MIDAS display relevant rm, im values for kernel
diagnostics
L X Do-it
L X Ld KERNEL                   ; load the microprogram
L B0 Addr RBASE 0
L B1 Addr RBASE 17
L B0 Val 0
L B1 Val 0
L B2 Addr MCR
L B2 Val  1              ; turn off stack overflow/underflow wakeups from memC
L B4 Addr STACKPTOPBITS
L B5 Addr STACKPADDR
L C0 Addr R0                    ; display common registers
L C1 Addr R1
L C2 Addr RM1
L C3 Addr R01
L C4 Addr R10
L C5 Addr RHIGH1
L C6 Addr RSCR
L C7 Addr RSCR2
L C8 Addr T 20
```

```
L C9 Addr FLAGS                 ; control for  hold, task simulator
L C10 Addr ITERATIONS           ; count of iterations
L C11 Addr NEXTTASK
L C12 Addr HOLDVALUE
L C13 Addr
L C14 Addr
L A19 Addr TASK 20    ; use task 0 as default
L A19 Val 0
L A7 Addr RBASE 20    ; use rbase 17 as default
L A7 Val 0
L X DisplayOn                           ; May 18, 1981  11:43 AM
L X TimeOut 10000
L X Call XORTASKCIRC()                  ; TURN ON TASK CIRCULATE
L X Skip 1
L X ShowError Timed out
;L X TimeOut 100000
;L X Go BEGIN
;L X Skip 1
;L X ShowError Timed out
```

l. Introduction

I will attempt to give some ideas about what to do if your Dorado fails one of the Microdiagnostics or Midas command files.

The first thing to run is Testall in a random pattern. You can run **GarageTestall** if you have a complete Dorado, or select **TESTALL** from the Midas comand menu. See **GarageTestAll.Midas** for a list of which board and page number in the logic each register or memory is located.

Another way to isolate a problem is to run the **SimTest** from the Midas command menu. After a failure use the middle button on the mouse and bug **DMux** to find which signal and board is causing the problem.

Note: after you center mouse button **DMux** the name changes to **DWrong**.

A sample of a typical DMux failure would be: bSwitch'a/A. Meaning the signal bSwitch'a was bad on the ContA board. ColVic.0/C would mean the signal ColVic.0 is bad on the MemC board.

/B=ContB
/A=ContA
/L=ProcL
/H=ProcH
/I=IFU
/C=MemC
/X=MemX
/D=MemD

You can find the exact value of the microinstruction that failed by pushing the **center** and **right** mouse button at the same time while the mouse is pointing at **DMux** or **DWrong**. Then point the mouse at the value of **MIR** and push the **Center** mouse button.

When any of the microdiagnostics fail, point the mouse at the value of OLINK 20 and push the center mouse button to find the routine that called the error. The labels are sometimes helpful in trying to figure out which part of the machine is being checked.

For example a label of "QrwErr" would indicate that the microdiagnostic failed the "Q" register during the read/write test. The "Q" register is located on the ProcL and ProcH boards. Another example is the label "SvarErr", this is a failure of the Storage board (MSA) during the MemA microdiagnostic. There is always some **was** and **should be** information to get you on the track to fixing the computer.

One thing to remember is that the Dorado was designed to be checked out from the bottom board to the top, but there is still a lot of interaction between all of the boards. For example a Dorado could fail the Kernel microdiagnostic which is supposed to check only the bottom 5 processor boards, but the board causing the failure is the DispY board.

1a. Instant-on Jumper:
There is a place on the Baseboard side panel to install a "Instant-on" jumper. (left side pins # 114 to 115). This will cause the Baseboard to turn the power on without the usual delay. It is only installed during troubleshooting and should be removed after the Dorado is repaired.

1b. Parity errors
Pe's from Input (IOB parity) errors come from any Input or Output controller (DispY, DispM, or Dsk/Eth) They are detected on the ProcL and ProcH board. To disable IOB parity errors type **IOBPE** to the Alto display and **Left** mouse button RdCmds. Bits 0-7 are detected of the ProcH board and bits 8-15 are detected on the ProcL board. See the section of tasks to figure which task is causing the problem.

ST (Storage Transport) parity errors are detected and generated on the MemD board.

PE'S FROM CACHE are parity errors from the CacheA located on the MemC board.

RAM Parity errors are caused by bad data from the R-memory on the ProcL and ProcH boards. The erorors are detected on the ProcL and ProcH boards. To disable RAM parity errors type **RAMPE** to the Alto display and left mouse button **RdCmds**. Bits 0-7 are on the ProcH board and bits 8-15 are on the ProcL board.

STK Parity errors are caused by bad data from the stack on the ProcL and ProcH boards. The errors are detected on the ProcL and ProcH boards. Bits 0-7 are on the ProcH board and bits 8-15 are on the ProcL board.

PES from T are caused by bad data from the "T" register located on the ProcL and ProcH boards. Bits 0-7 are on the ProcH board and bits 8-15 are on the ProcL board.

MesaFG parity errors are caused by bad data from the MemD board and detected on the IFU board.

Parity errors from IMX are caused by bad data on the ContB board.

I will attempt to list the boards to logically change first, but after that you can change boards by using the Bottom-Up or the Top-Down approach.

2. Kernel.midas

TestAll and Simtest should run successfully before looking into a failure of Kernel.

Kernel is the processor microdiagnostic, it checks the basic computer data paths.

Check the value of OLINK 20 to get an idea of where the problem is.
Remember bits 0-7 are the ProcH board and bits 8-15 are the ProcL board.

Change: ProcH, ProcL, ContA, ContB, and Base board.

3. MemA.Midas

TestAll, Simtest and the microdiagnostic Kernel should run successfully before looking into a failure in MemA.

MemA tests the MemC, MemX, MemD, and the MSA boards in that order. Usually the label for the test being run has the first letter that tells what board the microdiagnostic is trying to check. (C=MemC, X=MemX, D=MemD, and S=MSA).

Example: the label "cfBadCol" is for the MemC portion of the test. and XrwPipe3er is checking the MemX board. Remember center mouse button the value of OLINK 20 to get the label that called the error.

3a. Single bit errors.

An error at "SVARERR" is a failure of the MSA board. For single bit errors you can left mouse button **RdCmds**, then left mouse button **BADCHIP**. Midas will tell you the bad chip.

If there is a Double bit failure you will have to examine the failing munch and compare the data with the expected data. The expected data can be found on the Alto display listed as "SEXPECTED".

The label SEXPECTED is the expected data from the memory system.

The label RSCR2 is the actual data.
The label T20 is the bad bits.

3b. To display a failing Munch:

1. There are 3 entries listed for Pipe 0. Push the center mouse button while pointing at the second entry. We are looking for the FirstFaultSRN number.

2. Type: **Pipe 6** to the alto display (if the FirstFaultSRN is 6). Point the mouse at PIPE 0 and push the left mouse button. This will display the entry for Pipe 6.

3. We are looking for the address and Row in the CacheD where the failing munch is located. The row and address are displayed by pushing the center mouse button while pointing at the first entry of the PIPE 6 information.

4. Type ROW 20 (if ROW 20 was the failing pipe entry) to the Alto display, point the mouse at an unused area of the display and push the left mouse button.

5. There will be 5 entries for each row displayed. The first 4 are the address of the munches stored in that row and the 5th entry is the Victim and next Victim entries. We want to push the center mouse button on the Row entry that matches the address of the failing munch.

6. There will be 16 different 16 bit words displayed (1 Munch). Words 0, 2, 4, 6, 10, 12, 14, and 16 are the bottom MSA board. Words 1, 3, 5, 7, 11, 13, 15, and 17 are the top MSA board.

4. MemMisc.Midas

TestAll, Simtest and the microdiagnostics Kernel, and MemA should run successfully before looking into a failure of MemMisc.

MemMisc was created because MemA got too big. Most of the programs in MemMisc were orginally in MemA. When there is an error check the value of OLINK 20 to get an idea about which board is being tested and go from there.

5. IfuSimple.Midas

TestAll, Simtest and the microdiagnostic Kernel should run successfully before looking into a failure of IfuSimple.

IfuSimple checks the IFU board without using the any of the memory system. You can remove all of the boards above the IFU to run the test.
A failure in the IfuSimple microdiagnostic is almost always the IFU board.

Change: IFU, ContA, ProcH, ProcL.

6. IfuComplex.Midas

TestAll, Simtest and the microdiagnostics Kernel, IFUSimple, MemA, and MemMisc should run successfully before looking into a failure of IfuComplex.

IfuComplex requires all of the processor and the memory system installed to run the test. The Dsk/Eth and DispY boards are not used in this test. The IFU board (Instruction Fetch Unit) interacts with most of the boards in the Dorado. A failure of IFUComplex is usually the IFU board itself, but the rest of the Dorado is always suspect.

If the value of OLINK 20 is **opifAd20err**, the IFU board has sent the Dorado to the wrong location in the Instruction Memory. You can do **RdCmds** and then **Klink** to find out where the Diagnostic was before the IFU board sent the Dorado to never-never land.

During the running of IfuComplex the register Klink is loaded before every IFU jump instruction.

Change: IFU, MemD, ContA.

7. EventCounters.Midas

TestAll, Simtest and the microdiagnostics Kernel, IFUSimple, MemA, MemMisc, and IFUComplex should run successfully before looking into a failure of EventCounters.

EventCounters requires all of processor and memory system installed before running. The Event counters are located on the IFU board and are used to count certain conditions that happen in the Dorado. Things like IfuJumps, Hold conditions, and Cache misses can be counted for analysis. A Dorado that fails EventCounter will usually run the operating systems ok. (Cedar, SmallTalk, Mesa)

Change: IFU.

8. TriconD.Midas

TestAll, Simtest and the microdiagnostics Kernel, IFUSimple, MemA, MemMisc, and IFUComplex should run successfully before looking into a failure of TriconD.

TriconD checks-out a portion of the Dsk/Eth board. It does not check the actual paths to and from the Disk or the Ethernet. This is mainly a test of the Control portion of the board.

Change: Dsk/Eth, ProcH, ProcL. DispY.

9. Triex.Midas

This is a Program to test a T-300 disk on a Dorado. We should never get a microdiagnostic failure here.

10. KernelS.Midas

KernelS.Midas (speed test) first runs the microdiagnostic Kernel at 31 ns. After successful completion of one pass the clock speed is decreased by one nano-second and the test is run again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored. This is due to a known hardware problem. The command file will continue to speed up the Dorado until Kernel fails. A failure at speeds faster than 27 nano-seconds can usually be ignored.

See **Kernel.Midas**

11. MemaS.Midas

MemAS.Midas (speed test) first runs the microdiagnostic MemA at 31 ns. After successful completion of one pass the clock speed is decreased by one nano-second and the test is run again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored. This is due to a known hardware problem. The command file will continue to speed up the Dorado until MemA fails. A normal Dorado will always run faster than 28 ns. Since MemA requires about 6 minutes to make a full pass it should take a while before a failure happens.

See **Mema.Midas**.

12. MemMiscS.Midas

MemMiscS.Midas (speed test) first runs the microdiagnostic MemMisc at 31 ns. After successful completion of one pass the clock speed is decreased by one nano-second and the test is run again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored. This is due to a known hardware problem. The command file will continue to speed up the Dorado until MemMisc fails. A normal Dorado will always run faster than 28 ns.

See **MemMisc.midas**.

13. IfuSimpleS.Midas

IfuSimpleS.Midas (speed test) first runs the microdiagnostic IfuSimple at 31 ns. After successful completion of one pass the clock speed is decreased by one nano-second and the test is run again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored. This is due to a known hardware problem. The command file will continue to speed up the Dorado until IfuSimple fails. A normal Dorado will always run faster than 28 ns.

See **IfuSimple.Midas**.

14. IfuComplexS.Midas

IfuComplex at 31 ns. After successful completion of one pass the clock speed is decreased by one nano second and the test is run again. At the clock speed of 29 nano-seconds and faster, RM parity errors from the ProcH board are ignored. This is due to a known hardware problem. The command file will continue to speed up the Dorado until IfuComplex fails. A normal Dorado will always run faster than 28 ns.

See **IfuComplex.Midas**.

15. SboardTest.Midas

This Midas command file first loads the MemA microdiagnostic and then changes MemA to run the Storage board tests only. This is helpful when working on a known bad MSA board.

For single bit errors left mouse button **RdCmds** and then left mouse button **BadChip** to get the failing chip location. (See BadChip.Midas on page 19)

16. CboardTest.Midas

This Midas command file first loads the MemA microdiagnostic and then changes MemA to run the MemC board tests only. This is helpful when working on a known bad MemC board.

You need to ground the pin#109 on the left sidepanel of the MemC board to run this test if the boards above the MemC are removed.

17. XboardTest.Midas

This Midas command file first loads the MemA microdiagnostic and then changes MemA to run the MemX board tests only. This is helpful when working on a known bad MemX board.

18. DboardTest.Midas

This Midas command file first loads the MemA microdiagnostic and then changes MemA to run the

MemD board tests only. This is helpful when working on a known bad MemD board.

19. FIO.Midas

The FIO test has to have the FIO test Board installed in one of the fast I/O slots. This tests the Fast Input/Output of the Dorado.

Change: MemD, ProcH, ProcL.

20. LAG.Midas

Go to the microdiagnostic that failed to figure out the problem.

21. LAG1.Midas

Go to the microdiagnostic that failed to figure out the problem.

22. LAGSpeedtest.Midas

Go to the microdiagnostic that failed to figure out the problem.
A failure at speeds faster than 29 nano-seconds can usually be ignored.

23 KernelTask.Midas

KernelTask checks to see that the proper task switching is taking place during the **Task Circulate** phase of the **Kernel** microdiagnostic.

Change: ContA.

24. VoltageTest.Midas

A failure of the VoltageTest is either a bad power supply or Baseboard, be sure the voltages are set with a good digital volt meter before changing anything.

Change: Baseboard or the failing power supply.

25. CurrentTest.Midas

The current test checks for known current readings for a fully loaded Dorado. An error here is usually the Baseboard, but could easily be a loose connection on the power buss.

Change: Baseboard or tighten loose connections.

26. TemperatureTest.Midas

The Temperature test checks for certain limits on the boards that are supposed to have temperature sensors installed.

Change: The failing board, fans, check room temperature.

27. VIT.Midas

VIT calls the Voltage test, Current test, and the Temperature test.

28. TestAllGarage.Midas

This is a list of all of the registers or memories tested during **TestAllGarage**. The page number in the logic drawings is also listed. Sometimes a test failure can be caused by a board that is not supposed to be active during the test. A MemC board can cause a ContA type of failure. It is a good idea to open all of the edge connectors down to the minimum configuration and rerun the test. I have listed all of the mimimum board configurations to run each test.

"CPREG" (**C**ontrol **P**rocessor) pg. 6 thru 19 ContA Board (Baseboard?).
The CP register is 16 bits wide and shared between bits 0-7 and 8-15 of register CPOut. A failure of the CPOut register will show up as a 2 bit failure of the CP register. For example bits 0 and 8 of the CP register will fail when the problem is bit "0" of the CPOut register.
The minimum board configuration to test "CPREG" is: BaseBoard, ContB, and ContA.

"MIR" (**M**icro **I**nstruction **R**egister) pg. 1 ContA, pg. 1-2-3 ContB.
The minimum board configuration to test "MIR" is: BaseBoard, ContA, and ContB.

"Q" REG. Bits 0-7 pg.17 ProcH board, Bits 8-15 pg. 17 ProcL board.
The minimum board configuration to test "Q" is: BaseBoard, ContB, ContA, ProcL, and ProcH. A configuration of BaseBoard, ContB, ContA, and ProcL can be used but only bits 8-15 will be tested.

"CNT" (**C**ou**nT** Register) Bits 0-7 pg. 17 ProcH, Bits 8-15 pg. 17 ProcL board.
The minimum board configuration to test "CNT" register is: BaseBoard, ContB, ContA, ProcL, and ProcH. A configuration of BaseBoard, ContB, ContA, and ProcL can be used but only bits 8-15 will be tested.

"SHC" (**Sh**ift **C**ontrol) Bits 0-7 pg. 18 ProcH, Bits 8-15 pg. 18 ProcL board.
The minimum board configuration to test "SHC" is: BaseBoard, ContB, ContA, ProcL, and ProcH. A configuration of BaseBoard, ContB, ContA, and ProcL can be used but only bits 8-15 will be tested.

"MEMBX" (**Mem**ory **B**ase Inde**x**) pg. 24 ProcH board.
The minimum board configuration to test "MEMBX" is: BaseBoard, ContB, ContA, ProcL, and ProcH.

"STKP" (**St**ac**k P**ointer) pg. 25 ProcL board.
The minimum board configuration to test "STKP" is: BaseBoard, ContB, ContA and ProcL.

"TASK" pg. 26 ContA board.
The minimum board configuration to test "TASK" is: BaseBoard, ContB, and ContA.

"PROCSRN" (**P**rocessor **S**torage **R**eference **N**umber) pg. 3 MemX board.
The minimum board configuration to test "PROCSRN" is: BaseBoard, ContB, ContA, ProcL, ProcH, IFU, MemC, and MemX.

"MCR" reg (**M**emory **C**ontrol **R**egister) pg. 4, 5 MemC board.
The minimum board configuration to test "MCR" is: BaseBoard, ContB, ContA, ProcL, ProcH, IFU, and, MemC.

"INSSET" (**Ins**truction **Set**) pg. 4 IFU board.
The minimum board configuration to test "INSSET" is: BaseBoard, ContB, ContA, ProcL, ProcH, and IFU.

"EVENTB" (**Event** Counter **B**) pg. 14 IFU board.
The minimum board configuration to test "EVENTB" is: BaseBoard, ContB, ContA, ProcL, ProcH, and IFU.

"ESTAT" (**E**rror **Stat**us) pg. 9 ContB board.

The minimum board configuration to test "ESTAT" is: BaseBoard, ContB, and ContA.

"TPC" (**T**ask **P**rogram **C**ounter) pg. 6 thru 19 ContA board.
The minimum board configuration to test "TPC" is: BaseBoard, ContB, and ContA.

"TLINK" (**T**empory Link Register) pg. 6 thru 19 ContA board.
The minimum board configuration to test "TLINK" is: BaseBoard, ContB, and ContA.

"IMX" (**I**nstruction **M**emory) ContB board.
The minimum board configuration to test "IMX" is: BaseBoard, ContB, and ContA.

"IMBD" (**I**nstruction **M**emory **BD**UMX) pg. 12 ContB board.
The minimum board configuration to test "IMBD" is: BaseBoard, and ContB.
Note: If the ContA board in not installed you will have to left mouse button **Config**, then left mouse button **Control** to run the IMBD test.

"ALUFM" (**A**rithmetic **L**ogic **U**nit **F**unction **M**emory) pg. 11 ProcL (ALUFDEC).
The minimum board configuration to test "ALUFM" is: BaseBoard, ContB, ContA, and ProcL.

"T" (**T**emporary Storage) Bits 0-7 pg. 2-9 ProcH, Bits 8-16 pg. 2-9 ProcL boards.
The minimum board configuration to test "T" REG is: BaseBoard, ContB, ContA, ProcL, and ProcH. A configuration of BaseBoard, ContB, ContA, and ProcL can be used but only bits 8-15 will be tested.

"RBASE" (**R**egister memory **Base**) pg. 23 ProcL board.
The minimum board configuration to test "RBASE" is: BaseBoard, ContB, ContA, and ProcL.

"TIOA" (**T**ask **I**nput/**O**utput **A**ddress) pg. 23 ProcH board.
The minimum board configuration to test "TIOA" is: BaseBoard, ContB, ContA, ProcL and ProcH.

"MEMBASE" (**Mem**ory **Base**) pg. 25 ProcH.
The minimum board configuration to test "MEMBASE" is: BaseBoard, ContB, ContA, ProcL and ProcH.

"RM" (**R**egister **M**emory) Bits 0-7 pg. 2-9 ProcH, Bits 8-16 pg. 2-9 ProcL boards.
The minimum board configuration to test "RM" is: BaseBoard, ContB, ContA, ProcL, and ProcH. A configuration of BaseBoard, ContB, ContA, and ProcL can be used but only bits 8-15 will be tested.

"STK" (**ST**ack **M**emory) Bits 0-7 pg. 2-9 ProcH, Bits 8-16 pg. 2-9 ProcL boards.
The minimum board configuration to test "STK" is: BaseBoard, ContB, ContA, ProcL, and ProcH. A configuration of BaseBoard, ContB, ContA, and ProcL can be used but only bits 8-15 will be tested.

"BR" (Memory **B**ase **R**egister) pg. 1 and 7-10 of the MemC board.
The minimum board configuration to test "BR" is: BaseBoard, ContB, ContA, ProcL, ProcH, IFU, and MemC.

"CACHEA" (**Cache A**ddress) pg. 11 and 12 of the MemC board.
The minimum board configuration to test "CACHEA" is: BaseBoard, ContB, ContA, ProcL, ProcH, IFU, and MemC.

"CACHED" (**Cache D**ata) MemD board.
The minimum board configuration to test "CACHED" is: BaseBoard, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, and MemD.

"MAP" (Memory **Map**) pg. 12 and 13 of the MemX board.
The minimum board configuration to test "MAP" is: BaseBoard, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, and MemD.

"VM" (**V**irtual **M**emory) This tests uses the entire memory system. The micordiagnostics Mema and

MemMisc should be used to work on this type of failure.
The minimum board configuration to test "VM" is: BaseBoard, ContB, ContA, ProcL, ProcH, IFU, MemC, MemX, MemD, and 2 MSA boards..

"IFUM" (**I**nstruction **F**etch **U**nit **M**emory) pg. 6 IFU board.
The minimum board configuration to test "IFUM" is: BaseBoard, ContB, ContA, ProcL, ProcH, and IFU.

"SHMV" (**Sh**ifter **M**ask **V**ector) Bits 0-7 pg. 16 ProcH, Bits 8-15 pg. 16 ProcL boards.
The minimum board configuration to test "SHMV" is: BaseBoard, ContB, ContA, ProcL, and ProcH.

"WF" (Mesa **W**rite **F**ield) pg. 18 ProcH, pg. 18 ProcL boards.
The minimum board configuration to test "WF" is: BaseBoard, ContB, ContA, ProcL, and ProcH.

"RF" (Mesa **R**ead **F**ield) pg. 18 ProcH, pg. 18 ProcL boards.
The minimum board configuration to test "RF" is: BaseBoard, ContB, ContA, ProcL, and ProcH.

"ProcVA" (**Proc**essor **V**irtual **A**ddress) pg. 1, 7, 8, 9, and 10 MEMC board.
The minimum board configuration to test "ProcVA" is: BaseBoard, ContB, ContA, ProcL, ProcH, IFU, and MemC.

29. AcceptanceTests.Midas

Go to the individual test that failed to localize the problem.

30. SimTestNoErrors.Midas

Point the mouse at **DMux** and push the **center** mouse button to see the failing signal. Sometimes a perfectly good Dorado will get a few errors when running **SimTest**, but if the error is persistent you should look into the problem.

A sample of a typical DMux failure would be: bSwitch'a/A. Meaning the signal bSwitch'a was bad on the ContA board. ColVic.0/C would mean the signal ColVic.0 is bad on the MemC board.

/B=ContB
/A=ContA
/L=ProcL
/H=ProcH
/I=IFU
/C=MemC
/X=MemX
/D=MemD

31. SimTest20.Midas

Abort Midas and print the file **Simerror.Report** to see the 20 SimTest errors.

32. How to initialize Partition 19

This procedure will set-up Partition 19 (T-315 disk only) with the Smalltalk exerciser, Lisp exerciser, and all of the Alto files to run Bravo, Sil, and Laurel. (Partition 5 for T-80 disk)

To run the SmallTalk exerciser type: **@RunSmallTalk.cm** C/R (This is a free running program)

To run the Lisp exerciser type: **@RunLisp.cm** C/R (This is a free running program)

To run the SmallTalk color test type: **@Colortest.cm** C/R (See section 37 on how to run this test)

To get back to the Alto mode hold down the "M" key on the keyboard and push the boot button 3 times.

1. Boot from the net
2. type: **NewOS.boot** C/R
3. Do you want to install this Operating System?
Y
4. Do you want the long installation dialog?
Y
5. Do you want to ERASE a disk before installing?
Y
6. Type the name of the directory where Alto programs are kept.
IO
7. If the host is an IFS or Maxc, this should probably be "Alto"
If the host is just another random Alto, type <return>:
Alto
8. Use both of the disks?
Y
9. Use all 406 cylinders of the disk?
Y
10. Use all 14 sectors of the disk?
Y
11. When the disk is ready, type OK to proceed, A to abort:
OK (This will take about 1 minute)
12. Do you want to disable error logging through the net?
N
13. Do you want to change the error logging address (currently [3#5#30])?
N
14. Do you want to change memory error parameters?
N
15. What is your Name:
Type in your login name
16. Please give your disk a name:
Partition 19 (Partition 5 for a T-80 disk)
17. Do you wish to give your disk a password?
N
18. The computer will go into the FTP program to get the executive program for your new disk. At this time it will ask for your password.
(Type in your password)
19. Type: **Ftp Io Ret <Vest>NewDoradodisk.cm** C/R
20. Type: **@NewDoradodisk.cm** C/R (There will be about 5809 pages left on your disk when finished)

33. How to initialize Cedar

This procedure will initialize partition 1 thru 16 with Cedar and install the microcode on the disk. This procedure will take over an hour to complete. (Partition 1 thru 4 for a T-80 disk)

1. Boot from the net.

2. Type: **Cedar** C/R

3. Type: **Switches: N** C/R

4. Type: **Cedar** C/R

5. After about 15 seconds you will have to type in your name and password.

6. Do you want to initialize your disk from scratch for use by Cedar (Selecting from the standard options I will offer you)?
Type: **Y** (Type **N** to get to IAGO)

7. How many Alto partitions do you want to have on Drive RD0?
Type: **1** (Type **3** for a T-315 disk system, we will install LISP partititons for testing)

8. Has your disk already been formatted? (Confirm if the disk has previously been used for Cedar or Pilot, AND you have not reduced the amount of disk reserved for Alto volumes)
Type: **N**

9. Do you want to password-protect this disk?
Type: **N**

10. Do you want to have a disk world-swap debugger (CoCedar) on this disk?
Type: **N**

11. I intend to perform the following operations:
- format the entire disk (excluding the Alto disk regions);
- create a new physical volume named "The name of your computer";
- create a client logical volume named "Cedar" with a 20000 page virtual memory;
- install microcode, germ and boot files(s) from the release directory;
- Make the "Cedar" volume be your physical boot volume;
- boot the volume(s) to initialize the software.

This operation will destroy all files on the entire disk on drive RD0 (excluding the Alto disk regions).
Are you sure you want to do this?
Type: **Y**

After Cedar is initialized you can bring in some programs using the DF tool.

1. **[Ivy]<Jacobi>5.2>Top>DoradoBoard.df**
1a. You will have to type in: **Getboards.cm** to the command tool to get all of the files on your local disk to run the Displayboard program.
2. **[Indigo]<Cedar5.2>Top>CkViewer.df**
2a. After loading CkViewer type to the command tool: **cd ///Commands** C/R
2b. Type: CkViewer C/R
3. **[IVY]<Atkinson>5.3>RussHacks.df**
3b. Type: Vbounce C/R
You will need a machine with a color board (DispM) installed to run DoradoBoard and CkViewer. See page 44 for how to install the DispM board.

34. How to initialize DEMOLISP

1. Boot to the Net Executive and type: **Partition 18** C/R
2. Type: **NewOS.boot** C/R
3. Do you want to install this Operating System?
Y
4. Do you want the long installation dialog?
Y
5. Do you want to ERASE a disk before installing?
Y
6. Type the name of the directory where Alto programs are kept.
IO C/R
7. If the host is an IFS or Maxc, this should probably be "Alto"
If the host is just another random Alto, type <return>:
Alto C/R
8. Use both of the disks?
Y
9. Use all 406 cylinders of the disk?
Y
10. Use all 14 sectors of the disk?
Y
11. When the disk is ready, type OK to proceed, A to abort:
OK (This will take about 1 minute).
12. Do you want to disable error logging through the net?
N
13. Do you want to change the error logging address (currently [3#5#30])?
N
14. Do you want to change memory error parameters?
N
15. What is your Name:
Type in your login name
16. Please give your disk a name:
Partition 18 C/R
17. Do you wish to give your disk a password?
N
18. The computer will go into the FTP program to get the executive program for your new disk. At this time it will ask for your password.
(Type in your password)
19. Type: **Ftp IO Ret <Vest>MakedemoLISP.cm** C/R
20. Type: **@MakedemoLISP.cm** C/R
This will take about 5 minutes. There will be about 5200 pages left of your disk when finished.
21. After LISP is installed on the disk type: **LISP [IO]<EMS>Demo.Sysout** C/R
22. After LISP is up and running you can type in:
    1. **(CHANGEBACKGROUND TWODOLLAR)** (Note Capitol letters)
    2. **(CHANGEBACKGROUND TIFFANY)**
    3. **(CHANGEBACKGROUND STEINHEIM)**
    4. **(CHANGEBACKGROUND RHINE)**
    5. **(DEMO)** After about 5 minutes you will have to put up a window for the "Mirror Demo".
    6. You can run the standard LISP tests by typing **LOAD (TEST))**. After the load is finished type **(TEST)**.

    LISP COLORDEMO:
    1. **LOAD (DORADOCOLOR.DCOM))**
    2. **LOAD (COLORDEMO.DCOM))**
    3. Hit **Return** one time if the program stops loading.
    4. **(COLORDEMO)** A window will open up and you can run the color programs from there.

Note: to stop any of the test programs while they are running hold down the **CTRL** key and push the **D** key.

35. Task information

A common failure of the Dorado is for a task request to get stuck on all of the time. The symptoms are that the machine will not boot and when running the microdiagnostics you will get an error of: Failed to zero TOPE. TOPE (TO Priority Encoder) is a register that corresponds to the TWReq lines. You can look at the register TOPE by typing **TOPE** to the Midas Alto display. Point the mouse at an unused area of the screen and push the **left** mouse button.

For example if "TOPE" had a value of 10 that would indicate bit#12 or the Disk is requesting a task wakeup all of the time. The signal to look at would be "DiskTW" on the DskEth board or "TWReq.12" on the ContA board.

Task # Octal	Task # Decimal	Signal Name	Function
0	0	No name for task "0"	The Emulator.
1	1	TWReq.1	Special task for restarting emulator after faults.
2	2	TWReq.2	Junk task that is awakened every 32 us. (JunkTW)
3	3	TWReq.3	DispY horizontal task request. (WakeDHT)
4	4	TWReq.4	DispM horizontal task request. (WakeAHT)
5	5	TWReq.5	Not Used.
6	6	TWReq.6	Ethernet output task request. (WakeEthTx)
7	7	TWReq.7	Ethernet input task request. (WakeEthRx)
10	8	TWReq.8	Not Used.(For future 10 mb Ethernet board(WakeEthTx))
11	9	TWReq.9	DispM word task request. (WakeAWT)
12	10	TWReq.10	Task Simulator (TestTW).(Also for future 10 mb Ethernet board(WakeEthRx))
13	11	TWReq.11	DispY word task request. (WakeDWT)
14	12	TWReq.12	Disk I/O task request. (DiskTW)
15	13	TWReq.13	Not Used.
16	14	TWReq.14	Not Used.
17	15	TWReq.15	Fault task request. (TWReq15)

36. Task wiring information

Task wiring is a 30 gauge twisted pair wirewrap, with the black wire going to a ground nearest the signal wire.

Task # Octal	Task # Decimal	Signal Name	Wiring information on Right side panel
0	0	No name for task "0"	None
1	1	TWReq.1	Pin# 44 ContA
2	2	TWReq.2	Pin# 45 ContA to IFU pin# 13 (JunkTW)
3	3	TWReq.3	Pin# 48 ContA to DispY pin#121 (WakeDHT)
4	4	TWReq.4	Pin# 56 ContA to DispM pin#121 (WakeAHT)
5	5	TWReq.5	Pin# 57 ContA
6	6	TWReq.6	Pin# 60 ContA to DskEth pin#121 (WakeEthTx)
7	7	TWReq.7	Pin# 61 ContA to DskEth pin#120 (WakeEthRx)
10	8	TWReq.8	Pin# 64 ContA to 10mb Ethernet pin#121 (WakeEthTx)
11	9	TWReq.9	Pin# 128 ContA to DispM pin#120 (WakeAWT)
12	10	TWReq.10	Pin# 129 ContA to ProcH pin#109 (TestTW) Also the 10mb Ethernet board will be wired to this task from pin#120
13	11	TWReq.11	Pin# 132 ContA to DispY pin#120 (WakeDWT)
14	12	TWReq.12	Pin# 133 ContA to DskEth pin#117 (DiskTW)
15	13	TWReq.13	Pin# 136 ContA
16	14	TWReq.14	Pin# 137 ContA
17	15	TWReq.15	Pin# 140 ContA to MemX pin#132 (TWReq15)

37. SmallTalk Colortest

The SmallTalk colortest is a easy to run test for the DispM and DispY boards. You can get the test by loading from the dump file: [IO]<EMS>SmallTalkColortest.dm. If you are using Partition 19 that was initialized using the command file "NewDoradoDisk.cm" all of the files are already loaded.

The test is started by typing in @ColorTest.cm C/R

The test will come up with 16 green lines that represent the color green starting at the value of "0" and going to the value of "256 in steps of 16. You can left mouse button the blue color and red color to add them to the screen. Red, Blue and Green on at the same time will result in the color white to be displayed.

You can also change the **Steps** that the color is incremented by. The test starts at 16 but you can change to 8, 4, 2 or 1. When the color display is in the "1" step mode you will have to have a very dark room to see anything.

38. Grid test for DispY and DispM boards

The data path on the DispY and DispM boards starts out as a 16 bit path then goes to a 32 bit path and into a 32 bit serial shift register.

A bit failure in these data paths are shown on the black and white display as either a solid line from top to bottom of the display repeating itself every 32 bits, or an absence of data bits repeating itself overy 32 bits.

I have made a Sil drawing that shows all of the 32 bits represented as lines on the display. It is very easy to study the drawing and figure out which bit is in error and start debugging from there.

The sil drawing is stored at: [IO]<Vest>Grid.sil.

To display the file you need a disk with Sil installed and then type: **Sil.run Grid.sil** C/R

See page 45 for an example of a display showing bit#25 stuck at a "0".

See page 46 for an example of a display showing bit#0 stuck at a "1".

39. Special Baseboard Proms

I have two special Baseboard proms to use in the debugging of Baseboards. They are a single 2716 E-Prom chip installed in location B-10 of the baseboard. The other 3 E-Proms are not used during the running of this program. To get these proms send me a message and I will mail you a copy.

The one I find the most useful has a simple program running that stores a -1 into each different 6532 chip and than increments the "DAC" by one.
When scoping the Baseboard using this prom you can see each of the 6532 chips being selected (pin# 37) and a -1 being stored into each chip. Also by looking at the output of the "DAC" (j21.14) you can see a sawtooth waveform going from 0 volts to 2.56 volts.

The other prom has a program installed that allows you to DTACH to the Dorado and modify the program by changing the value at the location "PROBLEMS".

If the Value of "PROBLEMS" is equal to a "0" then: turn off the lite and spin.

If the Value of "PROBLEMS" is equal to a "1" then: increment the "DAC" from 0 to 2.56 volts.

If the Value of "PROBLEMS" is equal to a "2" then: turn all of the "DAC" bits on and then off.

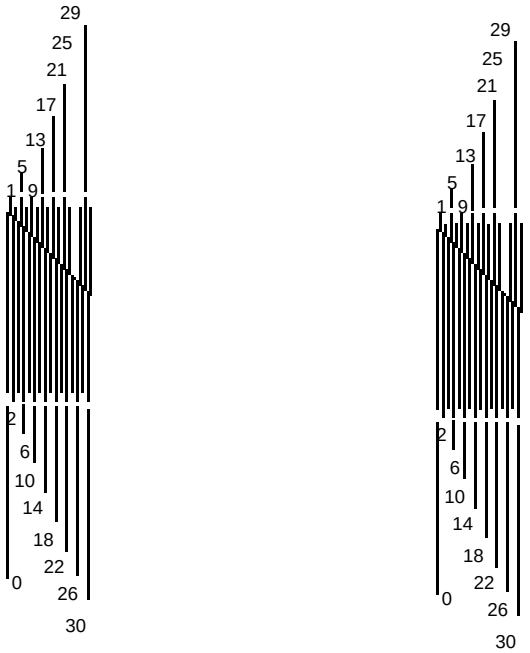
If the Value of "PROBLEMS" is equal to a "3" then: this routine will turn the lite on and if we exit midas the watch dog timer should reboot the dorado and turn off the lite.

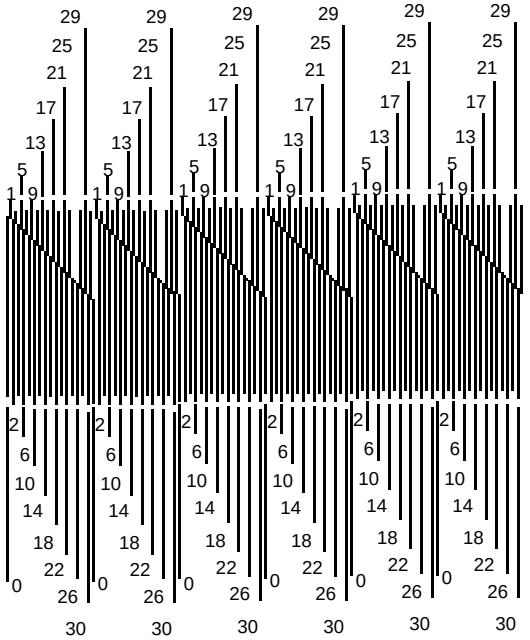
If the Value of "PROBLEMS" is equal to a "4" then: the program that you stored starting at location "E0" will be run.

40. How to install a DispM board

1. Turn off the Dorado.
2. Install a DispM board in slot 20 of the Dorado. Slot 20 is the location above the DispY board.
3. Move the cable block from the DispY board pins#144-175 on the right side panel to the DispM board pins#144-175 on the right side panel.
4. Remove the jumper from pins 180-181 on the right side panel of the DispY board.
5. Add a jumper from pins 180-184 on the right side panel of the DispM board.

This is an example of bit# 25 stuck to a "0".





This is an example of bit#0 at a "1" all of the time